FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Multi-Sensorial Biofeedback Augmented Horror Games

Gabriel Martins Souto

U. PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Pedro Gonçalo Ferreira Alves Nogueira

September 29, 2017

Multi-Sensorial Biofeedback Augmented Horror Games

Gabriel Martins Souto

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Doctor Jorge Barbosa

External Examiner: Doctor Luís F. Teófilo

Supervisor: Doctor Pedro Nogueira

September 29, 2017

Abstract

Throughout the history of video games, there has been an enormous evolution in their graphic capabilities. However, for a long the interaction with the player was only made with the same types of peripherals, namely with a mouse and keyboard or with a command. It was only decades later that other innovative devices started to appear, beginning with *Eye Toy* (2003), eventually expanding to other devices such as *Kinect* (2010), *Oculus Rift* (2013) and Biofeedback devices. The introduction of this type of devices has altered the potential of video games as it has allowed them to have a greater immersion capability. Biofeedback arose in the 1980s and it can be split in two types, direct or indirect. Direct Biofeedback, which requires conscious user participation (for example, by moving your body), knowing how your data affects the program, or indirect, that involves collecting data from the player without it (measuring heart rate or the skin's conductivity), the latter being of great interest because it is possible, from these sensors, to infer the player's emotions and to adapt the game itself to the player's emotional states, allowing a more personalised and realistic experience.
To make the experience as captivating as possible, different peripherals will be used in situations where they make more sense, that is, when an in-game action should be more difficult or a special skill.
These peripherals were be integrated in a horror game, having as a study case an experimental comparison of their performance with and without previous calibration. It was seen that most participants preferred to play the game with previous calibration. Nevertheless, a favourable opinion about both conditions was shown overall.

Resumo

Ao longo da evolução da indústria dos videojogos, notou-se uma enorme evolução na componente gráfica destes. No entanto, durante bastante tempo a interacção com o jogador foi só feita com os mesmos tipos de periféricos, nomeadamente com rato e teclado ou com um comando. Só mais tarde é que se começaram a desenvolver dispositivos mais inovadores em termos de interactividade como, começando pelo *Eye Toy* (2003), eventualmente expandindo-se com outros dispositivos como o *Kinect* (2010), os *Oculus Rift* (2013) e dispositivos de *Biofeedback*. A introdução deste tipo de dispositivos alterou o potencial dos videojogos, pois permitiu-lhes ter uma maior capacidade de imersão.

O *Biofeedback* surgiu na década de 80 e pode ser de dois tipos, directo ou indirecto. O *Biofeedback* directo, que requer uma participação consciente do utilizador (movendo, por exemplo, o seu corpo), sabendo este como os seus dados afectam o programa, ou indirecto, que envolve a recolha de dados do jogador sem este estar ciente de como isto irá afectar o programa (recolhendo-se, por exemplo, o ritmo cardíaco ou condutividade da pele), tendo este último tipo grande interesse, pois é possível, a partir de destes sensores, inferir as emoções do jogador e adaptar o jogo em si aos estados emocionais do mesmo de forma a fornecer uma experiência mais personalizada e realista.

De forma a tornar a experiência o mais cativante possível, foram usados periféricos diferentes e em situações em que façam mais sentido, isto é, em situações do jogo em que a acção deva ser mais difícil ou uma espécie de habilidade especial.

Estes periféricos foram integrados com um jogo de horror, tendo como caso de estudo uma comparação experimental entre as a performance dos mesmos com e sem calibração prévia. Notou-se que a maioria dos participantes preferiu a versão do jogo com calibração prévia. No entanto, uma visão favorável de ambas as versões foi demonstrada por todos os participantes.

Acknowledgements

First of all, I'd like to thank my supervisor for giving me the opportunity for working in this project and for all the help with the faced challenges. Secondly, to my family for all the support, and for being with me throughout all these years, I wouldn't be here without you. And, for last, to my colleagues and friends, who were by my side all these years. Thank you for discussions we've had. This wouldn't have been the same without you.

Gabriel Martins Souto

"I hear and I forget. I see and I remember. I do and I understand."

Confucius

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Abbreviations

EEC	Electroencephalography
EEG	Electroencephalogram
EMG	Electromyography
BP	Blood Pressure
BVP	Blood Volume Pulse
HR	Heart Rate
BP	Blood Pressure
BVP	Blood Volume Pulse
SC	Skin Conductance
EKG	Electrocardiogram
EDA	Electrodermal Activity

Chapter 1

Introduction

In this chapter we'll give an introduction about this thesis, by introducing its context, the motivation behind it and its goals.

1.1 Context

There are recordings of board games from as early as 4600 years ago . Since early times, mankind has used games for several purposes, from entertaining to preparing to war [ENST12]. Naturally, with new technology, it was only a matter of time until video games would appear, in the 1950's.
However, despite appearing so early, the evolution video games had until the 2000's was mostly graphical, in such way that video games, nowadays, have very impressive graphical qualities. Nevertheless, the peripherals used remained the same (keyboard/mouse or controller) until *Oshiete Your Heart*(1997),*Dance Dance Revolution* (1998) and *Eye Toy* (2003) appeared. Eye Toy detected motion with a camera, having its main interaction with the user this way, Dance Dance Revolution had a pad that allowed players to use their feet and *Oshiete Your Heart* accounted the player's BVP and SC into the gameplay. This started the dawn of a new generation of video games based on new types of interaction, such as *Kinect* (2010) and *Oculus Rift* (2013) [DDAM14].
However, even though the way people interact with video games has been diversifying over the last years, there is still an emotional barrier to overcome, where players are affected by games, but most video games do not adjust themselves to these reactions.
This dissertation is a follow-up previous study [NTR+15], where it was concluded that, in fact, biofeedback brings advantages to gameplay.

1.2 Motivation and Goals

Biofeedback gaming is being targeted by industrial giants such as Valve [Amb11], Ubisoft, Sony and Microsoft [NTR+15] to innovate the game industry.

The work of [NTR+15] is also a starting point for this thesis, having concluded that different mechanics and implementations affect the gameplay significantly. This leads to this thesis's objectives which are to:

- Calibrate the sensors in advance to gameplay
- Interpret the data provided by the sensors
- Calibrate the sensors during gameplay

1.3 Report Structure

Besides this introduction, this document includes another 5 chapters. In the 2nd chapter, we'll discuss current biofeedback technologies and their impact in video games. In the 3rd chapter we'll describe the developed videogame, followed by the presentation of the implemented biofeedback framework in the 4th chapter. In the 5th chapter we'll discuss our user experiment and its results. In the 6th chapter the final conclusion about this work will be presented.

Chapter 2

The Impact of Biofeedback in Video Games

This chapter will introduce the concepts of biofeedback, its research and the introduction of biofeedback in video games.

2.1 Biofeedback

Biofeedback started to appear in the 1960's, mostly from curiosity about how the mind and the body were connected. This led to the creations of devices the measure physiological sensors for the body, using them to improve health and performance of that people. Nowadays, Biofeedback is a mainstream technology in medicine, helping improve the quality lifestyle. As in the Association for Applied Psychophysiology and Biofeedback[1]:

"Biofeedback is a process that enables an individual to learn how to change physiological activity for the purposes of improving health and performance. Precise instruments measure physiological activity such as brainwaves, heart function, breathing, muscle activity, and skin temperature. These instruments rapidly and accurately "feed back" information to the user. The presentation of this information — often in conjunction with changes in thinking, emotions, and behavior — supports desired physiological changes. Over time, these changes can endure without continued use of an instrument."

2.1.1 Direct and Indirect Biofeedback

Biofeedback can be split in two kinds, Direct and Indirect Biofeedback [dSNR14]. In Direct Biofeedback, users are aware and in control of how their physiological data influences the feedback, using, for example, body movements.

[1] http://www.aapb.org/i4a/pages/index.cfm?pageid=3463

In Indirect Biofeedback, users are unaware of how to manipulate their data. The system will instead receive the player's data and work from there.

2.2 Arousal and Valence

Arousal and valence are very used to represent human emotions in biofeedback, pretty much separating human emotions in their energy level (arousal) and if they are a positive or negative emotion (valence) [Amb11].

2.3 Affective Feedback

Affective feedback has the objective of detecting emotions through different sources. This doesn't imply that it has to come through Biofeedback, as new technology is being researched to evaluate one's emotions from written posts and other social media [Cam16].
Nevertheless, a great affective feedback example occurs in [GATG17], where they measured several physiological responses of several participants in different game modes, solo, competitive and collaborative, concluding that collaborative gameplay brings up more positive responses and that competitive gameplay reveals more negative emotions. Other interesting features are suggested in [WFB16], to implement in physical devices the ability to send different combinations of incorporate thermal, vibrotactile and visual signals to help transmit human emotions.

2.4 Affective Gaming

The academic community refers to biofeedback in Video Games as Affective Gaming. The three heuristics proposed in [GDA05] are: “assist me”, “challenge me”, and “emote me”. The "assist me" heuristic is about how a video game should be able to detect a player's frustration and assist the player (for example, by supplying hints or easing the game difficulty). The "challenge me" heuristic is supposed to prevent the player from getting bored (by adjusting difficulty or triggering in-game events). Finally,

Biofeedback in video-games has started to emerge over the past decade with the Eye Toy in 2003, which had a camera that allowed the user to interact with the game with body movements. This led to a diversification of video-games in the market.
Even though there were great improvements in terms of Video Games and Biofeedback usage since 2003, most of its implementations were done with direct Biofeedback. Nevertheless, it was still an important step towards a more varied approach to gaming other than keyboard/mouse and controller.
There are, however, some of the early experiments with indirect biofeedback as well, such as [HB00], where two players would compete directly to see who would be more relaxed, using EEG. A more recent use of EEG in video games was *Mindlight* [SMLA+16], a serious game

made to help children cope with anxiety. The relaxation of the player would be mapped in game as a bright light that would fade away as the child became more stressed, being the only way for players to see in the dark haunted house.
Cardiovascular Measures have also been proven a valid method for Biofeedback in video games, with [DC07, LGR+16]. *Nevermind* in particular is very interesting in this aspect, because it is proof that a commercial implementation of this sort is possible, having some interesting game mechanics as well.
Another interesting implementation in is the implementation of *ChillFish* [SJ16], a serious game meant to calm children with ADHD that used a respiration sensor, with promising results. Respiration sensors were also used in [dSNR14], concluding that this sensor helped provide a lot more immersion when used as a direct biofeedback source.
A great example of a game using SK is seen in [PGO15], where they created a car game that would try its best the keep the player aroused, through different mechanics, weather, steering and speed. Its conclusion was that game difficulty and arousal were related in car games and that the mechanic that was changed affected the arousal in a different way.
It is also worth mentioning that Death Trigger [NM10] was one of the most complete games in terms of multi-modal implementation of Biofeedback, using SK, cardiovascular measures, EMG and a strain sensor.

"The future lies in 'affective gaming', that is games that will be 'intelligent' enough not only to extract the player's commands provided by speech and gestures, but also to extract behavioural cues, as well as emotional states and adjust the game narrative accordingly, in order to ensure more realistic and satisfactory player experience."[KZG+16]

Figure 2.1: Electroencephalogram Example[2]

2.5 Biofeedback Measures

2.5.1 Electroencephalography

Electroencephalography(EEC) uses an electroencephalogram (EEG) (Fig: 2.1) to measure a person's brainwaves in a noninvasive manner. This sensor is positioned on the scalp, where it will measure EEG signals, which are electrical signals recorded as multiple channels.

This sensors fits in the indirect feedback category because a one cannot control his brainwaves.

Two core research areas are present: new electrode technologies and lower power consumption electronics.

Different EEGs measure different wavelengths, being as different as having 4 channels or having up to 256 channels, according to how powerful the EEG is [CYS+10]. A great use of this kind of feedback is shown in [SOH15], where through the computation of 8 EEG channels, they were able to estimate the mood of a person and the adequate music for their mood.

Some of the most recent EEGs are wearable [CYS+10] that, unlike unwearable EEGs are more discreet, aesthetic and cheaper.

EEC is a very good method to measure the arousal and valence of a person. However, it is noisy and difficult to validate.

[2] www.neurosky.com

Figure 2.2: Electromyogram Example[3]

<

2.5.2 Electromyography

EMG measures the electrical impulses produced by a muscle to analyse its state [Her96], through an Electromyogram (Fig: 2.2) placed upon this muscle.
This sensor is easier to implement using direct biofeedback (by voluntarily moving a muscle), although it can be used in indirect biofeedback as well (by measuring involuntary muscle contractions).

Facial EMG can be used successfully to distinguish valence [NRON13].

2.5.3 Cardiovascular Measures

Cardiovascular measures are among one of the most used types of indirect biofeedback in video games and it has been proven useful in detecting emotions [DC07]. These measures are values obtained of regarding to the cardiovascular system, which concerns the heart. There are a variety of these measures and it is possible to obtain these values through various methods, such as an Electrocardiogram or a Blood Pressure sensor. HR is easy to measure, cheap and easy to obtain the arousal of a person. However, a person cannot move a lot and it is harder to obtain the valence of a person.

[3] http://www.biometricsltd.com

Figure 2.3: Electrodermal Sensor Example[4]

2.5.4 Electrodermal Activity

It's possible to obtain a person's Electrodermal Activity (EDA) through an Electrodermal sensor (Fig. 2.3). It's easy to obtain the index of arousal through this measure, usually in the for of skin conductance (SC). There are specific sweat glands located in the palms of the hands and soles of the feet that respond to psychological stimulation (e.g., cold and clammy hands when nervous)t [MN16].

[4] https://www.movisens.com

Figure 2.4: Temperature Sensor Example[5]

2.5.5 Temperature Sensors

A temperature sensor can be used as a physiological sensor when applied to a person's skin. Since body temperature is not something that's a human person can change willingly, it is related to indirect biofeedback. A great advantage of this sensor is that it is accessible.

[5] https://www.mindfield.de

Figure 2.5: Strain Sensor Example[6]

2.5.6 Strain sensors

A strain sensor is stretched across an individual's chest to measure its strain due to breathing volume (Fig. 2.5). Strain sensors can be directly controlled, as a person is able to control his/her breath. It can provide interesting data in terms of direct biofeedback to increase realism, as it was shown in [dSNR14], where a player had to held his breath when playing underwater.

2.5.7 Other Devices

Biofeedback has also seen an evolution to fit the demands of sport activities in order to enhance performance [LLL17], or help with other activities. It is interesting to notice that, while the most common use of the above mentioned is to have a feedback in a visual form, such as a number or a graph, it is also possible to supply feedback in other forms, such as in *PossessedHand* [TMR11], where it was possible to aid people learn to play musical instruments by using small electrical impulses to make the person's hand move.

[6] http://www.data-harvest.co.uk/

Chapter 3

The Developed Game

In order to test the biofeedback framework correctly, a videogame had to be developed, which will be described in this chapter. This chapter is divided in two main sub-sections: In section 3.1 the description of the game environment will be made. Next, in the sub-section 3.2 the enemy characters will be defined, followed by the description of the game progression in section 3.3. The Player character is presented in section 3.4, followed by the implemented mechanics in section 3.5. This chapter is concluded by presenting game UI in section 3.6 and the game menu in section 3.7.

3.1 Architecture

In order to develop a videogame that uses biofeedback mechanics, a specific biofeedback driven architecture had to be developed. As hown in Fig. 3.1, the main class in the architecture is the EmotionManager class. This class is responsible for reading and calculating the Arousal-Valence pair (and the slow motion mechanic), which will be used by every class that uses biofeedback mechanics. It is also worth mentioning that the Player's position is used by the EnemyAI and the EnemySpawner classes.

3.2 Environment

Since this is a horror game, it a dark forest environment with low luminosity was chosen, seen from the first perspective. The game map was relatively big, so some care was needed to make a clear path that the player should follow.

3.3 Enemies

The game has different kinds of enemies, spawning and behaving in different ways.

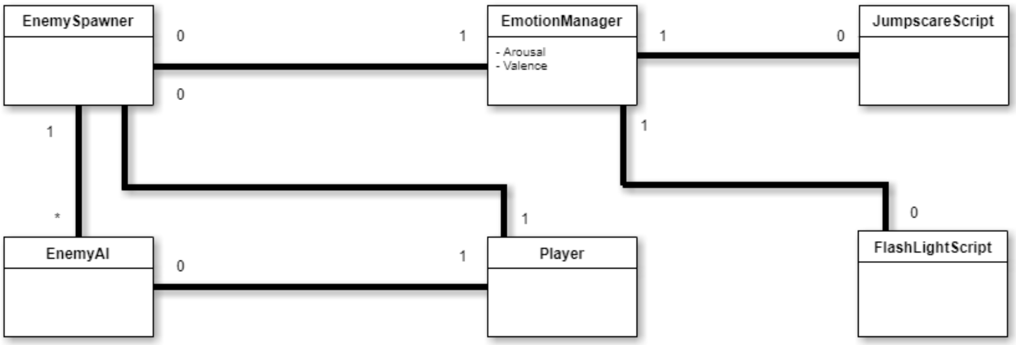

Figure 3.1: Game Architecture

3.3.1 Creeper

The Creeper (Fig. 3.2) is an enemy that dies with one shot from the player. Depending on the player's Arousal, the number of creepers will change at a time. The Creeper will walk slowly towards the Player, stopping and attacking the player if it gets within close range. Each attack deals 1 damage to the player (Listing 3.1). If the enemy gets too far away from the creeper, its speed will be increased greatly .

```
void FixedUpdate()
    {
        if (Vector3.Distance(transform.position, Player.transform.position) <
            attack_distance )
        {
            Player.TakeDamage(1);
        }
        else
        {
            if (agent.speed > 3.5f)
                anim.SetTrigger("Run");
            else
                anim.SetTrigger("Walk");
            agent.destination = Player.transform.position;
        }



    }
```

Listing 3.1: Creeper AI

3.3.1.1 Spawning

When the Game starts, there are no creepers instantiated. Instead, the C# class *EnemySpawner* is responsible for instantiating every creeper. This class will count the current instances of creepers in game, if they are fewer than the target number, it will spawn a new creeper. The spawning of an enemy happens with a delay up to 5 seconds and no other creeper can spawn during this offset. When the game starts, the initial target number is one, meaning that the first creeper will spawn up to 5 seconds in game. The spawning of the enemy creeper will occur within a minimum radius of 10 units and up to 28.284 units from the player. The target number will increase or decrease according to the players arousal(Listing 3.2).
This class is also responsible for adjusting the creepers' attack and movement speeds throughout the game, implemented with the Code seen in Listing 3.3.

3.3.1.2 Death

When the player shoots the creeper, the creeper dies instantly.

```
switch(EMF.Arousal)
        {
            case 0:
                enemyMax += 2;
                AdjustSpeed(true);
                break;
            case 1:
                enemyMax++;
                break;
            case 2:
                break;
            case 3:
                if(enemyMax > 1)
                    enemyMax--;
                break;
            case 4:
                if (enemyMax == 2)
                    enemyMax--;
                else if (enemyMax > 2)
                    enemyMax -= 2;
                AdjustSpeed(false);
                break;

        }
```

Listing 3.2: Creeper Number Adjustment

```
 private void AdjustSpeed(bool increase)
```

```
    {
        if (increase)
        {
            if (timeBetweenAttacks <= minAttackSpeed)
                timeBetweenAttacks = minAttackSpeed;
            else
                timeBetweenAttacks *= 0.75f;

            if (walkSpeed >= maxWalkSpeed)
                walkSpeed = maxWalkSpeed;
            else
                walkSpeed *= 1.25f;
        }
        else
        {
            if (timeBetweenAttacks >= maxAttackSpeed)
                timeBetweenAttacks = maxAttackSpeed;
            else
                timeBetweenAttacks *= 1.25f;

            if (walkSpeed <= minWalkSpeed)
                walkSpeed = minWalkSpeed;
            else
                walkSpeed *= 0.75f;
        }

    }
```

Listing 3.3: Creeper Speed Adjustment

3.3.2 Infested Horse

The Infested Horse (Fig. 3.3) is not exactly an enemy, as it can neither attack the player nor be attacked by the player, merely starting to follow if the player gets within 20 units.

3.3.2.1 Spawn

This character spawns near the third key, standing still until the player gets closer.

3.3.2.2 Death

This character won't die to the player's bullets. However, it will die if the player tries to interact with it, which will also lead to the spawn of 3 Spiders (Fig. 3.5).

3.3.3 Spiders

Each Spider (Fig. 3.4) behaves in a very similar way to the creeper, following and attacking the player (Listing 3.1). However, its initial movement and attack speeds are higher. The spider also deals 1 damage to the player per attack.

3.3.3.1 Spawning

These characters spawn in the horse's previous position.

3.3.3.2 Death

As with the creepers, the spiders die instantly.

3.3.4 Skeleton Guardian

The Skeleton Guardian (Fig. 3.6) is an enemy that is responsible for guarding second key. Unlike the other enemies, it doesn't die immediately, but only after loosing 10 hit points. This enemy has two main behaviours:

- Patrol
- Attack

3.3.4.1 Spawning

This character spawns near the second key.

3.3.4.2 Patrol

This is the starting behaviour of the Skeleton Guardian, where it justs roams between three points close to the key. This behaviour will continue until either the player attacks or enters the guardian's following range (20 units).

3.3.4.3 Attack

Once the guardian is damaged or the player gets within range, the guardian will start chasing the player. If it manages to get within its attacking range (5 units), it will attack the player with powerful hit, dealing 3 damage.

3.3.4.4 Critical Hit

It is possible for the player to take a critical hit against this enemy. Every time the player hits the enemy, a random number between 0 and 20 is generated. If this number is 20, a critical hit happens. A critical hit will give twice the damage and a different hit animation from the boss (Listing 3.4).

```
int crit = Random.Range(1, 21);
        if (crit > 19)
        {
            health -= 2;
        }
        else
        {
            health--;
        }
```

Listing 3.4: Critical Hit

3.3.4.5 Death

When the Guardian's hit points reach 0, it dies. Upon its death, the gate to the second key is opened (Listing 3.5).

```
    if (health < 1)
        {
            door1.ChangeDoorState();
            door2.ChangeDoorState();
        }
```

Listing 3.5: Skeleton Guardian's Death

3.4 Game Progression

The game starts with the Player facing the first key and a gate. There are two paths the player can follow, the path behind the gate and a path to the left. The path to the left will lead to the second key and the guardian. Tha path behind the gate will lead to a second junction, again left or front. The left path comes from the previous left path. The front path leads leads to the Infested Horse and the third key and, later the house, where the player will need the 3 keys win the game.

3.5 Player

3.5.1 Controls

The controls for the game are the w,a,s,d keys for movement, space-bar for jumping, the mouse for controlling the camera and taking aim, the mouse's left button to shoot, f to interact, the strain sensor for slow motion, and escape button to pause,.

3.5.2 Hit Points

The player started the game with a total of 10 hit points, displayed by a bar in the UI. These hit point would reduce with enemy attacks. If the player reached 0 hit points, the player would loose the game.

3.5.3 Interactions

The player is able to perform the following interactions (Listing 3.6):

- **Gates**: The player can open and close the first gate
- **Keys**: The player is able to pick up the keys throughout the map
- **Horse**: The play may interact with the Infested Horse
- **Front Door**: The player is able to open the house's front door

```
if(Input.GetButtonDown("Interact"))
      {
Ray ray = new Ray(transform.position, transform.forward);
RaycastHit hit;
if(Physics.Raycast(ray, out hit, interactDistance))
  {
    if(hit.collider.CompareTag("Door"))
    {
      DoorScript doorScript = hit.collider.transform.parent.GetComponent<
          DoorScript>();
      if(doorScript == null) return;

      if(doorScript.name == "hinge")
      {
        doorScript.ChangeDoorState();
        return;
      }
    }
else if(hit.collider.CompareTag("Key"))
    {
      Inventory.keys[hit.collider.GetComponent<Key>().index] = true;
      Destroy(hit.collider.gameObject);
                keyCounter++;
                keyText.text = "Keys: " + keyCounter + " / 3";
            }
else if(hit.collider.CompareTag("NextLeveldoor"))
    {
    if (keyCounter == 3)
                    manager.gameWon = true;

```

```
        }
        else if (hit.collider.CompareTag("Horse"))
                {
                    hit.collider.transform.parent.GetComponent<HorseAI>().Die();
                }
            }
    }
```

Listing 3.6: Interaction Script

3.6 Biofeedback Mechanics

There are 5 biofeedback mechanics. With the exception of the slow motion mechanic, all of the mechanics are indirect biofeedback mechanics.

3.6.1 Flashlight

The game is naturally dark, with the main source of light being the player's flashlight. So, one of the key mechanics was to reflect the player's arousal through this flashlight. The more aroused the player is, the more light the flashlight will provide, which will help the player with more visibility and making the game easier (Fig. 3.7). On the other hand, if the player is less aroused, the light will be smaller, providing a more challenging game experience (Fig. 3.8, Listing 3.7).

```
private void BiofeedBackUpdate()
    {

        timeCounter += Time.deltaTime;


        switch (EM.Arousal)
        {
            case 4:
                if (flashDesiredAngle == maxLightAngle)
                    break;
                timeCounter = 0;
                flashPreviousAngle = flashLight.spotAngle;
                flashDesiredAngle = maxLightAngle;
                break;
            case 1:
                if (flashDesiredAngle == lowLightAngle)
                    break;
                timeCounter = 0;
                flashPreviousAngle = flashLight.spotAngle;
                flashDesiredAngle = lowLightAngle;
                break;
```

```
            case 0:
                if (flashDesiredAngle == minLightAngle)
                    break;
                timeCounter = 0;
                flashPreviousAngle = flashLight.spotAngle;
                flashDesiredAngle = minLightAngle;
                break;
            default:
                if (flashDesiredAngle == highLightAngle)
                    break;
                timeCounter = 0;
                flashPreviousAngle = flashLight.spotAngle;
                flashDesiredAngle = highLightAngle;
                break;
        }


        if (flashDesiredAngle != flashLight.spotAngle && timeCounter <= changeSpeed)
        {

            flashLight.spotAngle = Mathf.LerpAngle(flashPreviousAngle, flashDesiredAngle, timeCounter);

        }

    }
```

Listing 3.7: Flashlight

3.6.2 Spawning of enemies

Similar to the flashlight, the spawning of enemies is also adjusted according to the player's Arousal, increasing the number of enemies active with low arousal and decreasing it with high arousal (Listing 3.2).

3.6.3 Jumpscare

The Jumpscare features the appearance of a frightening image to frighten the player. This feature was implemented in 2 different ways:

3.6.3.1 Incremental Method

Every time the the user has either low arousal, very high valence, or displays high boredom and frustration, a counter is incremented. When this counter reaches a certain value, the jumpscare occurs (Listing 3.8).

```
void incrementalMethod()
    {
        if(( EMF.Arousal < 2 || EMF.Valence >=4 || EMF.Boredom >=3 || EMF.
            Frustration >= 3 || EMF.Fun >=4) && !playing)
        {
            timeCounter += Time.deltaTime;

            if (timeCounter > TimeBetweenJumpScares)
            {
                scareSound.Play();
                scareImage.color = Color.white;
                playing = true;
                timeCounter = 0;
            }
        }
    }
```

Listing 3.8: Incremental Method

3.6.3.2 Probabilistic Method

As with the previous method, whenever the user shows low arousal, high valence, frustration or boredom, a Random float between 0 and 1 is created. If this number is bigger than 0.9, the jumpscare occurs (Listing 3.9).

```
void ProbabilityMethod()
    {
        timeCounter += Time.deltaTime;

        if (timeCounter > TimeBetweenJumpScares && !playing)
        {
            timeCounter = 0;

             if (EMF.Arousal == 0 || EMF.Valence == 5 || EMF.Boredom >= 4 || EMF.
                 Fun == 5)
            {
                //Probability of JumpScare
                if (Random.Range(0, 1) > 0.9)
                {
                    scareSound.Play();
                    scareImage.color = Color.white;
                    playing = true;
                }
            } else if (EMF.Arousal == 1 || EMF.Valence == 4 || EMF.Boredom == 3 ||
                EMF.Fun == 4)
            {
```

```
                //Probability of JumpScare
                if (Random.Range(0, 2) > 1.7)
                {
                    scareSound.Play();
                    scareImage.color = Color.white;
                    playing = true;
                }
            }
        }
    }
```

Listing 3.9: Probability Method

3.6.4 Slow motion

Whenever the player would hold his breath, the game would begin a slow motion effect, making it easier to shoot the enemies (Listing 3.10).

```

        if (EM.FullBreath)
            Time.timeScale = slowMotion;
        else
            Time.timeScale = 1;
```

Listing 3.10: Slow Motion

3.6.5 Enemy speed

In a similar way to the Spawning of enemies, the enemies' attack and movement speeds would also change according to the player's arousal, increasing with very low arousal and decreasing with very high arousal (Listing 3.3).

3.7 Game UI

The game had a simple UI to help guide the player:

- **Key counter** - This counted the number of keys the player had picked up and showed the number of total keys.
- **Life Bar** - This allowed to show the player its health
- **Aim** - This helped the player taking aim to take the shots
- **Creeper counter** - This counted the number of creepers the player had killed

3.8 Game Menu

The game had a simple Menu (Fig. 3.9) that appeared when starting the game. This menu was the same for pausing, when loosing or winning, the main difference between these being the play button, that would change its tag to "Resume" when the game was paused, and to "Play Again" when the game ended. Also, in the last two cases, the menu would show a "You win" or "You have died" message accordingly.

Figure 3.2: Creeper

Figure 3.3: Infested Horse

Figure 3.4: Spider

Figure 3.5: Spiders

Figure 3.6: Skeleton Guardian

Figure 3.7: Flashlight with high Arousal

Figure 3.8: Flashlight with low Arousal

Figure 3.9: Starting Menu

Chapter 4

Biofeedback Framework

This chapter will help to dive into the developed framework (Fig. 4.1). This framework is divided into smaller components, which will be discussed in detail in each of this chapter's sub-sections.

4.1 Biotrace+ Software

The Biofeedback sensors used were from the NeXus-10 device [1]. Biotrace+, the software provided by the NeXus-10, reads and processes the data provided by the sensors and can display the data according to the active channels. This software also calculates several values based on the primary channels (such as HR, HR variance, ...). It is possible to access this data in real time through a binary file.

4.2 Calibration

The calibration is a very important part of this framework, as every person will have different data read by different sensors, reacting in different ways to different stimuli, etc. This implementation stored an array with the maximum and minimum value for each channel, which was used by the rest of the framework.
As the data was calibrated during different phases, the following two scenarios had to be implemented.

4.2.1 Previous Calibration

The previous Calibration is done in a *Windows Forms* C# application in order to be used independently from the game. This application reads the sensor's live data from the Biotrace+'s binary file, saves the extreme values as a text file and the whole read data as a csv file.

[1] http://www.mindmedia.info/CMS2014/products/systems/nexus-10-mkii

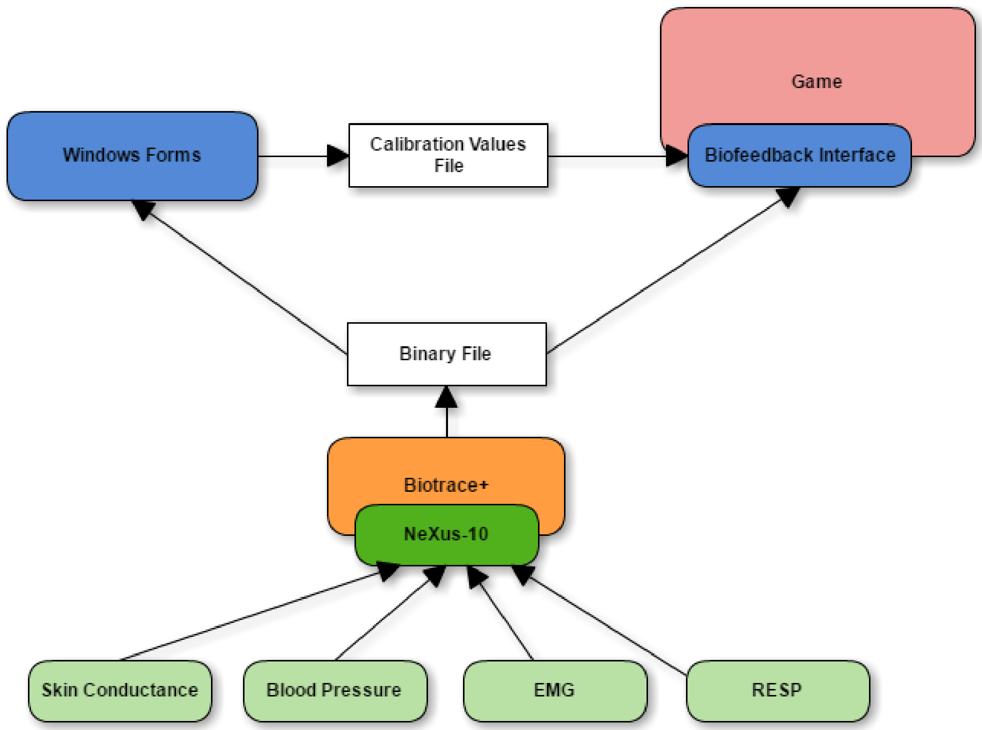

Figure 4.1: Architecture

4.2.2 Continuous

The continuous calibration can read the data from the previous mentioned text file, as well as the live data. Throughout the gameplay, it will constantly update the extreme values, as well as writing every calculated value into a csv file.

4.3 Implemented Fuzzy Logic

In order to convert the biological data into a pairs of Arousal and Valence, a Fuzzy Logic implementation was used, based in [MA07].

First, the data would be read from the filestream provided by the Biotrace+ Software. Then, the data was normalised to transform it from its raw values into a percentage(Listing 4.1). After this was done, the percentage was quantified using the same Fuzzy Logic functions from [MA07]. As seen in Listing 4.2, the GSR percentage was transformed into the belonging values for GSR Low, GSR MidLow, GSR MidHigh and GSR High. Once this was done, a random number would be used to get the Qualitative value for each sensor, as show in Listing 4.3. Lastly, the Arousal and Valence pair could finally be obtained.

```
private double Normalise(double raw, int index)
    {
        //The sensor might always read a new extreme value
        if (raw < minValues[index])
            minValues[index] = raw;
        if (raw > maxValues[index])
            maxValues[index] = raw;

        double range = maxValues[index] - minValues[index];

        double percentage = (raw - minValues[index]) / range;

        return percentage * 100;
    }
```

Listing 4.1: Normalise Function

```
case 0: //GSR

                ///GSR_LOW
                if (percentage <= 30)
                {
                    processedArray[(int)Sensor.GSR][(int)Value.Low] = -(percentage
                        - 30) / 30;
                }
                else
                {
                    processedArray[(int)Sensor.GSR][(int)Value.Low] = 0;
                }

                //GSR_MidLow
                if (percentage >= 20 && percentage <= 60)
                {
                    if (percentage < 40)
                    {
                        processedArray[(int)Sensor.GSR][(int)Value.MidLow] = (
                            percentage - 20) / (40 - 20);
                    }
                    else
                    {
```

```
                        processedArray[(int)Sensor.GSR][(int)Value.MidLow] = -(
                            percentage - 60) / Mathf.Abs(40 - 60);
                    }
                }
                else
                {
                    processedArray[(int)Sensor.GSR][(int)Value.MidLow] = 0;
                }


                //GSR_MidHigh
                if (percentage >= 40 && percentage <= 80)
                {
                    if (percentage < 60)
                    {
                        processedArray[(int)Sensor.GSR][(int)Value.MidHigh] = (
                            percentage - 40) / (60 - 40);
                    }
                    else
                    {
                        processedArray[(int)Sensor.GSR][(int)Value.MidHigh] = -(
                            percentage - 80) / Mathf.Abs(60 - 80);
                    }
                }
                else
                {
                    processedArray[(int)Sensor.GSR][(int)Value.MidHigh] = 0;
                }


                //GSR_HIGH
                if (percentage >= 60)
                {
                    if (percentage > 80)
                        processedArray[(int)Sensor.GSR][(int)Value.High] = 1;
                    else
                        processedArray[(int)Sensor.GSR][(int)Value.High] = (
                            percentage - 60) / (80 - 60);
                }
                else
                {
                    processedArray[(int)Sensor.GSR][(int)Value.High] = 0;
                }
                break;
```

Listing 4.2: Quatify GSR

```
//GSR
```

```
        double gsr_sum = processedArray[(int)Sensor.GSR][(int)Value.Low] +
            processedArray[(int)Sensor.GSR][(int)Value.MidLow] + processedArray[(
            int)Sensor.GSR][(int)Value.MidHigh] + processedArray[(int)Sensor.GSR][(
            int)Value.High];
        float belong_gsr = Random.Range(0, (float) gsr_sum);

        gsrLow = processedArray[(int)Sensor.GSR][(int)Value.Low];
        gsrMidLow = processedArray[(int)Sensor.GSR][(int)Value.MidLow];
        gsrMidHigh = processedArray[(int)Sensor.GSR][(int)Value.MidHigh];
        gsrHigh =  processedArray[(int)Sensor.GSR][(int)Value.High];

        if (belong_gsr < processedArray[(int)Sensor.GSR][(int)Value.Low])
            gsr = (int) Value.Low;
        else if (belong_gsr < processedArray[(int)Sensor.GSR][(int)Value.Low] +
            processedArray[(int)Sensor.GSR][(int)Value.MidLow])
            gsr = (int) Value.MidLow;
        else if (belong_gsr < processedArray[(int)Sensor.GSR][(int)Value.Low] +
            processedArray[(int)Sensor.GSR][(int)Value.MidLow] + processedArray[(
            int)Sensor.GSR][(int)Value.MidHigh])
            gsr = (int) Value.MidHigh;
        else
            gsr = (int) Value.High;
```

Listing 4.3: Qualitative Labeling

Chapter 5

Tests and Result Analysis

In order to verify the differences between gameplay with previous calibration and gameplay without previous calibration, tests were conducted with the participants.

5.1 Participants

The data were recorded from 11 healthy higher education students. From those 11 participants, 10 (90,9%) were male and 1 (9,1%) was female. Their age ranged between 19 and 25 (22.36364, 1.3666). Of the participants, 45.5% played videogames up to 4 hours a week, 27.3% from 4 to 8 hours, 9.1% from 8 to 16 hours, and 18.2% played more than 16 hours a week. When asked if they enjoyed playing horror videogames, 63.6% answered affirmatively, even though only 18.2% played this kind of games on a regular basis.

5.2 Procedure

Each test followed the following procedure

Briefing Before the test started, each participant had a small briefed over how the whole experimental session would go. Afterwards, the sensors would be set up in the following manner:

- SC was measured at the subject's right index and middle fingers using two Ag/AgCL surface sensors snapped to two Velcro straps.
- BVP was measured using a clip-on sensor on the left thumb.
- Facial EMG was measured with electrodes at the zygomaticus major (cheek) and the corrugator supercilii (eyebrow) muscles on the left side of the face.
- The RESP sensor was measured at the player's diaphragm using a Velcro strap.

After the setup, the sensors' behavior would be verified, by checking the pulse, if the respiration sensor was following the player's breath, and if the EMG sensors were responding to the players facial expressions.

Calibration The calibration process consisted in watching 2 videos:

- Comedy Clip: The participant saw a sketch from Monty Python's Flying Circus, the Dead Parrot Sketch.
- Tragedy Clip: The participant saw the final scene from Schindler's List

This allowed the framework to update the upper and lower boundaries for the current subject, saving them in a text file, to be used in the gaming phase later.

Game Once all the calibration data was recorded, each participant would then proceed to start playing. In this experiment, there were 2 conditions available: With Previous Calibration and Without Previous Calibration. As the name suggests, the difference between the two conditions was the use of the data gathered during the calibration phase. The order of the conditions changed from participant to participant to avoid habituation results.

Forms At the end of the experiment, the volunteering participants were asked to fill a form regarding the experiment.

5.3 Testing Environment

The developed videogame ran in unity in a laptop with the following specifications:

- Intel Core i7-3610QM CPU @ 2.30GHz
- 6GB RAM
- Nvidia GeForce 650M
- Windows 10 Education 64-bit
- 17' monitor

5.4 Results

In the analysis of the results both the questionnaires and the sensor's data were used to evaluate if the players noticed any difference between playing with and without previous calibration.

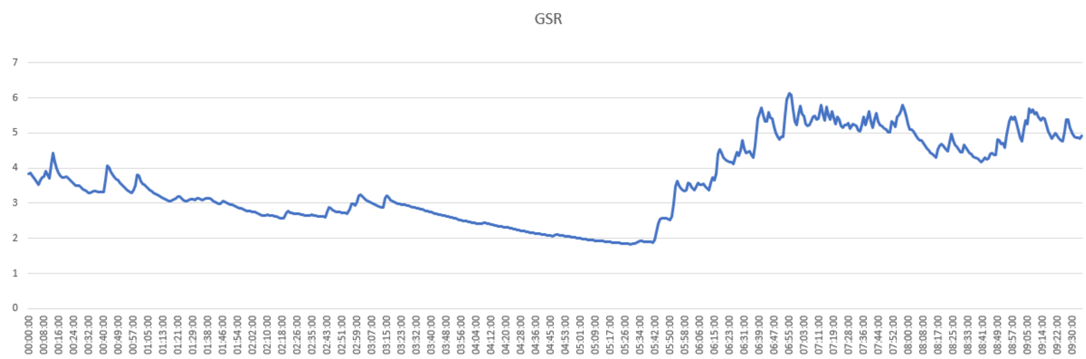

Figure 5.1: Participant 8's GSR

5.4.1 Calibration Analysis

The biological responses from each participant to the videos shown varied considerably between participants. This is to be expected, and there are some interesting conclusions, but there are some common points seen. Most participants (example: Participant 8, Fig. 5.1) had higher GSR values during the second video (which suggests higher arousal), with a clear spike in the moment where the video begins. This is also seen by a change in valence in most participants (zygomaticus major values are greater during the first video and corrugator supercilii values are greater in the second video, as seen with participant 8, Fig. 5.2).

5.4.2 Arousal Variance

From the data gathered, F-Tests and t-Tests were run over the dataset of arousal during each player's run. These tests were also used to compare the averages of the players with and without calibration. However, no statistical significance was found in this case (F = 1.1251, F Critical one-tail = 3.1373).

On the cluster of tests (Table 5.1), it is possible to see that in most participants there was a statistical difference seen between the arousal of the run with previous calibration the that of the run without previous calibration, with the exception of subject 3, where the t-test failed (t-Stat = -0.68446, t critical = 1.96048).

This fits with the user's responses to the question if they noted difference between the two types of gameplay, where a player only a player answered negatively.

5.4.3 Valence Variance

Even though the Valence was implemented in the same way as the Arousal, its resulting data isn't as believable as the Arousal results. This is because no significant changes were seen in any of the tests.

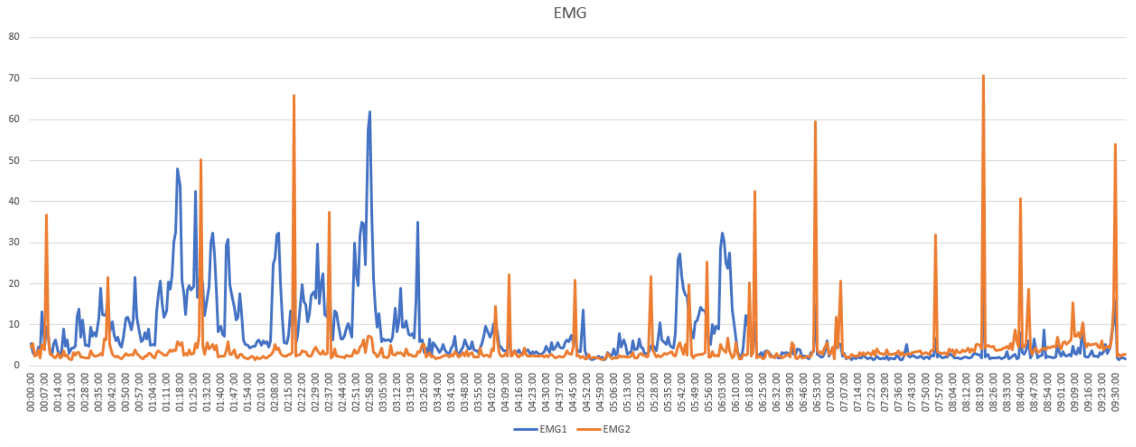

Figure 5.2: Participant 8's EMG

5.4.4 Game Mechanics

The game mechanic most favoured by the participants was the flashlight adjusting to their arousal level, with 72.7% picking this one. The second most preferred was the number of enemies adjusting to the Player's with 63.3% of the votes and the other mechanics tied with 45.5% each. This may be due to the fact that these were the most easily detected mechanics, when taking into account the fact that some participants had difficulties triggering the slow motion.

5.4.5 Player Preference

As expected, most players (63.5%) enjoyed more playing with previous calibration. The remaining players, however were evenly divided between preferring to play only the in-game calibration and no preference between the calibrations. This suggests that, even though only 1 person did not notice or experience the effects of previous calibration, there was another player that did notice them, but had no preference nevertheless.

Table 5.1: F-tests and t-tests on Participant Arousal

Participant	F-value	F Critical one-tail	t-Stat	t critical two-tail
1	1.8629804339504978	1.0740897261910765	37.71115604194128	1.9605355045716688
2	3.0142249772146457	1.0752048780568575	-22.470892306171955	1.9606472797142069
3	1.9104039470273892	1.0699862861817078	-0.68445662287965792	1.9604792688553374
4	3.946905480842994	1.0835972135109035	33.440151344901111	1.9607679234474296
5	5.3005305546459951	1.0623351554656015	-42.44269366529857	1.9602892735774775
6	1.2835248016938225	1.0791522410311631	19.661758782889461	1.9609403040412114
7	1.2252988036895778	1.0626635909827442	-6.3748325534889441	1.9603717446972824
8	1.5399004087508192	1.0587173351361527	10.502971158491937	1.9603226148159965
9	1.2779069534567309	1.0731248817515429	2.8878933845439652	1.9606200674175536
10	10.845187048386535	1.0722639320213601	-17.826875703929534	1.9613703496016655
11	3.3568648866093969	1.0771917518652445	-52.820757349531455	1.9609065550571663

Chapter 6

Conclusion

6.1 Goals

This work had the following objectives, to create a framework capable of calibrating the player's physiological data, the ability to interpret it, calibrate it in run-time and to test the effects of the previous calibration.
The first objective was completed with the creation of the Windows Forms application. The second objective, however, had some limitations. Even though the player's appeared to be enjoying the game, it's hard to believe their valence levels, although the rest of the results fit in with the literature. The last objective was the most interesting one. Even though the expected result was to see the player's preference for using previous calibration, this was not the case for everyone. This means that even though most people feel that the use of previous calibration as way to getter better gameplay, it's absence does not lead to a bad gaming experience.
Overall, this work can be considered a success, confirming that previous calibration is useful for these kinds of works.

6.1.1 Future Work

There is much future work that can be developed in this area. First of all, a better implementation of Valence should be done. Even though it is not the most important aspect in the range of a horror game, its information can always be useful.
It is also worth noting that more game mechanics can always be though, some which may be more interesting, or at least different somehow. A great example could be to use the EMG in other muscles of the body, such as the arm of legs, and use a direct biofeedback mechanic instead of an indirect one. It would also be interesting to use machine learning to predict the player's reaction to certain stimuli based on previous player profiling.

References

[Amb11] Mike Ambinder. Biofeedback in gameplay: How valve measures physiology to enhance gaming experience. In *game developers conference*, volume 2011, 2011.

[Cam16] Erik Cambria. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107, 2016.

[CYS+10] Alexander Casson, David Yates, Shelagh Smith, John Duncan, and Esther Rodriguez-Villegas. Wearable electroencephalography. *IEEE Engineering in Medicine and Biology Magazine*, 29(3):44–56, may 2010.

[DC07] Andrew Dekker and Erik Champion. Please biofeed the zombies: enhancing the gameplay and display of a horror game using biofeedback. *Proc. of DiGRA*, pages 550–558, 2007.

[DDAM14] Parth Rajesh Desai, Pooja Nikhil Desai, Komal Deepak Ajmera, and Khushbu Mehta. A review paper on oculus rift-a virtual reality headset. *arXiv preprint arXiv:1408.1173*, 2014.

[dSNR14] Goncalo A. da Silva, Pedro A. Nogueira, and Rui Rodrigues. Multimodal vs. unimodal biofeedback in videogames: An empirical player study using a first-person shooter. In *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*. Institute of Electrical and Electronics Engineers (IEEE), jun 2014.

[ENST12] Simon Egenfeldt-Nielsen, Jonas Heide Smith, and Susana Pajares Tosca. *Understanding Video Games: The Essential Introduction*. Routledge, New York, NY, 10001, 2nd edition, 2012.

[GATG17] Daniel Gábana Arellano, Laurissa Tokarchuk, and Hatice Gunes. *Measuring Affective, Physiological and Behavioural Differences in Solo, Competitive and Collaborative Games*, pages 184–193. Springer International Publishing, Cham, 2017.

[GDA05] Kiel Gilleade, Alan Dix, and Jen Allanson. Affective videogames and modes of affective gaming: assist me, challenge me, emote me. *Proc. of DiGRA*, 2005.

[HB00] Sara Ilstedt Hjelm and Carolina Browall. Brainball-using brain activity for cool competition. In *Proceedings of NordiCHI*, volume 7, 2000.

[Her96] Lee Herrington. EMG biofeedback: What can it actually show? *Physiotherapy*, 82(10):581–583, oct 1996.

[KZG+16] Irene Kotsia, Stefanos Zafeiriou, George Goudelis, Ioannis Patras, and Kostas Karpouzis. Multimodal sensing in affective gaming. In *Emotion in Games*, pages 59–84. Springer Nature, 2016.

[LGR+16] Adam Lobel, Marientina Gotsis, Erin Reynolds, Michael Annetta, Rutger C.M.E. Engels, and Isabela Granic. Designing and utilizing biofeedback games for emotion regulation: The case of nevermind. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 1945–1951, New York, NY, USA, 2016. ACM.

[LLL17] Rui Li, Daniel T. H. Lai, and Wee Sit Lee. *Biofeedback Technologies for Wireless Body Area Networks*, pages 659–686. Springer International Publishing, Cham, 2017.

[MA07] Regan L. Mandryk and M. Stella Atkins. A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies. *International Journal of Human-Computer Studies*, 65(4):329–347, apr 2007.

[MN16] Regan L Mandryk and Lennart E Nacke. Chapter 6 biometrics in gaming and entertainment technologies. In *Biometrics in a Data Driven World: Trends, Technologies, and Challenges*, pages 191–224. CRC Press, 2016.

[NM10] Lennart E Nacke and Regan L Mandryk. Designing affective games with physiological input. In *Workshop on Multiuser and Social Biosignal Adaptive Games and Playful Applications in Fun and Games Conference (BioS-Play)*, 2010.

[NRON13] Pedro Alves Nogueira, Rui Amaral Rodrigues, Eugénio C Oliveira, and Lennart E Nacke. Guided emotional state regulation: Understanding and shaping players' affective experiences in digital games. In *AIIDE*, 2013.

[NTR+15] Pedro A. Nogueira, Vasco Torres, Rui Rodrigues, Eugénio Oliveira, and Lennart E. Nacke. Vanishing scares: biofeedback modulation of affective player experiences in a procedural horror game. *Journal on Multimodal User Interfaces*, 10(1):31–62, dec 2015.

[PGO15] Avinash Parnandi and Ricardo Gutierrez-Osuna. A comparative study of game mechanics and control laws for an adaptive physiological game. *Journal on Multimodal User Interfaces*, 9(1):31–42, 2015.

[SJ16] Tobias Sonne and Mads Møller Jensen. Evaluating the chillfish biofeedback game with children with adhd. In *Proceedings of the The 15th International Conference on Interaction Design and Children*, IDC '16, pages 529–534, New York, NY, USA, 2016. ACM.

[SMLA+16] Elke A Schoneveld, Monique Malmberg, Anna Lichtwarck-Aschoff, Geert P Verheijen, Rutger CME Engels, and Isabela Granic. A neurofeedback video game (mindlight) to prevent anxiety in children: A randomized controlled trial. *Computers in Human Behavior*, 63:321–333, 2016.

[SOH15] Ryosuke Sawata, Takahiro Ogawa, and Miki Haseyama. Human-centered favorite music estimation: EEG-based extraction of audio features reflecting individual preference. In *2015 IEEE International Conference on Digital Signal Processing (DSP)*. Institute of Electrical and Electronics Engineers (IEEE), jul 2015.

[TMR11] Emi Tamaki, Takashi Miyaki, and Jun Rekimoto. Possessedhand: Techniques for controlling human hands using electrical muscles stimuli. In *Proceedings of the*

SIGCHI Conference on Human Factors in Computing Systems, CHI '11, pages 543–552, New York, NY, USA, 2011. ACM.

[WFB16] Graham Wilson, Euan Freeman, and Stephen Brewster. Multimodal affective feedback: Combining thermal, vibrotactile, audio and visual signals. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, ICMI 2016, pages 400–401, New York, NY, USA, 2016. ACM.

REFERENCES

Appendix A

A.1 Questionnaire

Quanto tempo despende semanalmente em video jogos?

○ 0-4 horas

○ 4-8 horas

○ 8-16 horas

○ Mais de 16 horas

Gosta de jogos de terror?

○ Sim

○ Não

Costuma jogar jogos de terror?

○ Sim

○ Não

Notou diferença entre jogar com ou sem calibração?

○ Sim

○ Não

Gostou mais de jogar com ou sem calibração?

○ Com calibração

○ Sem calibração

○ É Indiferente

Que mecânicas achou mais interessantes?

☐ Lâmpada ajustar-se ao jogador

☐ Número de inimigos ajustar-se ao jogador

☐ Jumpscare ajustar-se ao jogador

☐ O encolher da barriga causar câmara lenta

☐ A velocidade dos inimigos ajustar-se ao jogador